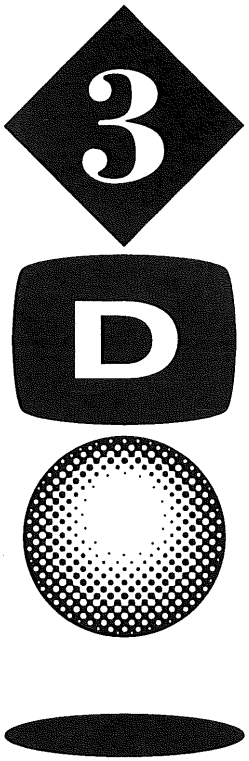
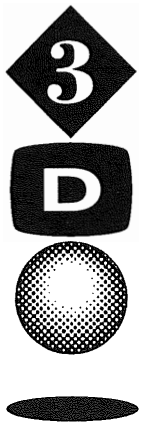


3DO M2 RELEASE ◆ VERSION 2.7

Release 2.7

- *3DO M2 Release 2.7 Release Notes*
- *3DO M2 Mercury Programmer's Guide*
- *3DO M2 Command List Toolkit*
- *3DO M2 Link/Dump Programmer's Guide*

03083-001



3DO M2 Release 2.7 Release Notes

Version 2.7 – June 1996

Table of Contents

Preface

About This Document

This document describes technical changes, bug fixes, and known problems in the 2.7 release of the 3DO M2 tools and operating system. This document should be used in conjunction with the *3DO Release 2.0 Release Notes*.

The *Release 2.7 Release Notes* contains the following information:

- All warnings, caveats, and known bugs that apply to the product at the Release 2.7 level. This includes warnings, caveats, and known bugs that were documented for Release 2.0 and are still applicable in Release 2.7.
- Features that were added or changed since Release 2.0.
- Bugs that were fixed since Release 2.0.
- Notes specifically for Release 2.7.

The *Release 2.7 Release Notes* DO NOT CONTAIN the following information:

- Supplementary documentation and tutorial-type information that was included in the *Release 2.0 Release Notes*. See the *Release 2.0 Release Notes* for that information.
- Features that were added or changed or bugs that were fixed in Release 2.0. See the *Release 2.0 Release Notes* for that information.
- Notes relating to Release 2.0. See the *Release 2.0 Release Notes* for that information.

The *Release 2.7 Release Notes* appear in the form of a hardcopy document and also in the form of an ASCII document on the distribution media. The hardcopy version may be slightly more recent. Any differences between the hardcopy and electronic versions of this document are indicated by change bars in the hardcopy version.

In addition to this document, there are component-specific release notes for some components in the form of ReadMe files on the distribution media. Read this document and the ReadMe files carefully because they contain important caveats and warnings that can make the difference between your code running or failing.

Note: *The hardcopy version of this document is more recent than and takes precedence over the electronic version and the ReadMe files.*

About This Release

This 3DO M2 2.7 release is an update to the 2.0 release and provides a number of enhancements and bug fixes.

This release includes the following items:

- M2_2.7 Software CD
- *3DO M2 Release 2.7 Read This First* document
- *3DO M2 Release 2.7 Release Notes*
- Some additional and replacement manuals for the 3DO M2 Developer's Documentation Set that was issued with Release 2.0. The complete document set was not reissued for Release 2.7.

  The new and replacement manuals provided with Release 2.7 are as follows:

  - *3DO M2 Mercury Programmer's Guide*
  - *3DO M2 Command List Toolkit*
  - *3DO M2 Link/Dump Programmer's Guide*

Audience

This document is for developers preparing titles for the M2 Development System.

Organiza

- **Preface**

  Brief description of this document.
- **Introduction to 3DO M2 Release 2.7**

  Notes about the release as a whole.

  This section contains some or all of the following subsections:

  - Introductory paragraph(s).

    Brief overview or description of this release.
  - READ THIS NOW!

    Particularly urgent information about this release.
  - Major Features Added or Changed

    Particularly important features added or changed in this release.
  - Major Bugs fixed.

    Particularly important bugs fixed in this release.
  - Major Caveats and Known Bugs

    Particularly important known bugs or things to watch out for in this release.
  - Notes

    Additional information about this release.
- **Installing 3DO M2 Release 2.7**

Notes on installing this release. This section contains some or all of the following subsections concerning the installation process:

- Introductory paragraph(s).
- READ THIS NOW!
- Features Added or Changed
- Bugs fixed.
- Caveats and Known Bugs
- Notes

- **Sections on individual components of the 3DO M2 system** (e.g., Hardware, C Development Tools, Graphics, Video and Data Streaming, Audio).

  Each section contains subsections about subcomponents. Each section or subsection contains some or all of the following subsections relating to the overall component or individual subcomponent:

  - Introductory paragraph(s).
  - READ THIS NOW!
  - Features Added or Changed
  - Bugs fixed.
  - Caveats and Known Bugs
  - Notes

- **Appendices**

  Additional information.

Typographical Conventions

The following typographical conventions are used in this book:

Item	Example
code example	`Scene_GetStatic(scene)`
procedure name	`Char_TotalTransform()`
new term or emphasis	In M2, *characters* are objects that can be displayed on the screen.
file or folder name	It is located in `"folder:subfolder"`.

1.0 Introduction to 3DO M2 Release 2.7

Release 2.7 of the 3DO M2 Portfolio and Toolkit consists of new, updated, and improved software and documentation.

The following sections contain information about the release as a whole. Information about individual components of the M2 system are contained in subsequent sections.

1.1 READ THIS NOW!

- **CDs created using Release 2.0 will NOT run on systems running Release 2.7.**
- **M2 Release 2.7 runs on Rev G or later dev cards. It does not run on earlier dev cards.**
- The complete document set was NOT reissued for Release 2.7. See "About This Release" on page vii in the Preface of this document for a list of the new and replacement manuals that are included in M2 Release 2.7:
- Setting the Current Directory

  The current directory of a program is now set by the system to refer to the directory where the executable being run is located. That is, the current directory of a program may be different than the current directory of the shell used to invoke the program.

  This behavior is intended to make it easier to write programs that don't rely on knowledge of their location. All of the files for a program can be accessed relative to the program's initial current directory, since it is assured that the current directory will always point to where the application is.

  This behavior can be defeated by specifying the `-Hflags=1` command-line argument when linking your programs. This causes the linker to set a bit in the executable that tells the system to initially set that program's current directory to be the same as that of the shell it was launched from. This behavior is what a normal shell environment typically does, and is desirable for any shell utility programs, such as `"ls"`.

1.2 Major Features Added or Changed

- Mercury Rendering Engine

  This release of M2 includes the Mercury Rendering Engine, a 3D graphics pipeline that is optimized to run in the 4K instruction cache of the PowerPC (PPC602) CPU that drives M2.

  Although Mercury is designed to be used by programs written in C or C++, it runs at a speed approaching that of assembly language because all of its internal procedures are written in assembly language and pass arguments by means of registers.

  Mercury is offered as an alternative to the original M2 Graphics Framework and Graphics Pipeline development packages, which are more traditional, more programmer-friendly but slower-running C-language APIs. Developers who want to create games that run at the fastest possible speed should consider using the Mercury API. Mercury is documented in the *3DO M2 Mercury Programmer's Guide*, which is also new with this release.

2.0 Installing 3DO M2 Release 2.7

This section describes changes to the installation process.

2.1 Notes

2.1.1 Post-installation Setup - Before Running the Debugger

To be able to run the debugger on an **8 Mb dev card,** you must do the following:

1. Copy the file "`ROM.m2.flash.debug`" into your dev card's Flash ROM. You must use the FlashROMTool to do this.
   - Run the **FlashROMTool**.
   - In the window titled "Onboard Flash", click in the Filename box.
   - Select the file "`ROM.m2.flash.debug`".
   - Click the Flash button.
   - When you see the "Verified" message, quit the FlashROMTool.

   You only need to do this once. You do not need to perform this first step again unless you receive a new OS release from 3DO or you use the FlashROMTool to copy another file into your dev card's flash ROM.
2. Set up the debugger.
   - Run the debugger. You must use debugger version 2.1a10 or later.
   - Under the Target menu, select Setup.
   - Make sure the ROM is set to "Debugger".
   - Click on the Script button and select the file "**debugger.flash.scr**".

To be able to run the debugger on an **16 Mb dev card,** you must do the following:

1. Set up the debugger.
   - Run the debugger. You must use debugger version 2.1a10 or later.
   - Under the Target menu, select "Setup".
   - Make sure the ROM is set to "Debugger".
   - Click on the Script button and select the file "**debugger.16M.scr**".

3.0 Hardware

3.1 Caveats and Known Bugs

- Flashing Boot Screen

  When booting version G of the 3DO Development Card with 3DODebug or Comm3DO, the 3DO M2 boot screen will flash. This problem is related to the video output hardware, and is confined to the boot screen. It will cease once you launch an application and will not affect your work.
- The following bug is known to occur with all development cards and could cause visual artifacts noticeable to developers using this development system.

  LOD determination can be incorrect for thin triangles.

  In very thin (narrow) triangles, some spans will not have any pixels drawn on them, making the lines appear dashed. This is expected. The problem is that the vertical LOD logic is not ignoring these lines for the purposes of

generating VLodLatch and VLodCompare signals. So a VLodCompare signal is being generated to compare against a previous line's VLodLatch, except that previous line had no pixels in it.

The result is that the LOD determination made by the Texture Mapper based on horizontal and vertical u,v deltas may be incorrect for these triangles.

When a texture is displayed at extreme perspective (as on the side of a cube), the image exhibits blockiness in the vertical direction. This means that you see visible banding when looking at a vertical column of pixels, while a horizontal row of pixels looks fine.

The cause has to do with the LOD determination performed by the Triangle Engine. The triangle engine looks at the texel delta in both the horizontal and vertical direction, and chooses the coarser LOD. In the case described above, there is significant compression of the texture in the horizontal direction, but little compression in the vertical direction. This means that the LOD is determined by the horizontal delta, with the effect that, vertically, a coarser LOD is used than is desired. This means that several successive vertical pixels will sample the same texel in the coarser LOD, resulting in blockiness in the vertical direction.

The same effect may be seen at the top of the cube, if the top face is just barely visible, but this time the vertical compression drives the LOD choice, and the blockiness appears in the horizontal direction.

There is no way to avoid these problems, but generally when triangles are so thin as to cause this to occur, the texture will already be so warped (such as on the side of a barely visible cube) that the overall degradation of the image should be slight.

4.0 C Development Tools

4.1 Compiler

4.1.1 Caveats and Known Bugs

- If you abort a compilation using Command-., the temporary asm files in "{TempFolder}" are not deleted. You must manually delete these files to avoid filling up your disk.

  Note that "{TempFolder}" is the folder "RAM Disk:temp:" if that folder exists. Otherwise, "{TempFolder}" is "{Boot}System Folder:Preferences:MPW:TempMPW:".

4.2 ANSI C Support

4.2.1 READ THIS NOW!

Strict ANSI checking does not allow C++ style comments "//". To allow such comments, do not specify the dcc flag -Xstrict-ansi.

4.2.2 Caveats and Known Bugs

- Floating point exceptions are not ignored by default. If you are using floating point numbers, or routines that use floating point numbers (e.g., Graphics, or Audio routines), you should consider installing a floating point exception handler. The default action for floating point exceptions is to terminate the task, which may not be desirable.

- The routine "`sqrtff`" does not handle infinities very well. If your math calculations could involve an infinity, you may need to install an exception handler.

4.3 Link3DO

4.3.1 Caveats and Known Bugs

- Under certain circumstances, `Link3DO` relocates assembly language branches incorrectly.

  The linker shipped on the M2_2.0 CD (version 1.15) relocates assembly language branches using truncated offset values under the following conditions:

  - You are programming in assembly language -- e.g., building or using a custom version of the Mercury library
  - You use a relative branch (such as `bge` or `bgel`) to an imported symbol which, at link time, is more than 32K bytes away from the branch.

  `Link3DO` 1.15 will not detect the problem and will relocate the branch using a truncated offset value. This can cause a crash or other type of error because of a wild run-time branch.

  This problem will only affect a small number of developers. If the problem affects you, try the new "experimental" linker, which fixes this problem. To do this,

  - Locate your current (i.e., M2_2.0) `Link3DO` by typing "`which Link3DO`" in your MPW worksheet and pressing enter. Your `Link3DO` should be in your 3DO folder with a path such as `HardDisk:3DO:tools:M2_2.0:MPWTools:Link3DO`.
  - Rename this linker "`Link3DO.orig`" and drag the new (i.e., M2_2.7) `Link3DO` into the same folder.
  - Before building your project again, quit and re-launch MPW.
  - You should then do a clean (i.e., "full") build of your project.

  The new linker will report an out-of-range error, if one is present, and give you information that allows you to correct it.

4.4 3DO Debug

4.4.1 Caveats and Known Bugs

- On a Power Macintosh, selecting the menu item "`File/Directories/Current Source`" will cause a crash under some circumstances.

  The circumstances leading to the crash are quite common (e.g., a "`.spt`" `script` has been run), and result from the default setup created when you use `CreateM2Make`. We recommend that you avoid selecting this menu item (`File/Directories/Current Source`).

- In the Target Setup dialog, the check box item "Initial bp in boot code" should never be selected. This functionality is broken, and will cause your target system to hang during boot.
- In the Source window, the PC arrow will only show up if it is at the first instruction of a source statement. Because of a bug in the line information, the PC arrow will occasionally disappear when source stepping. This most commonly occurs when stepping through "`while`" and "`for`" loops.

- Source files larger than 32K will not display correctly in the Source window.
- The debugger has trouble switching between development cards. If you have two or more cards in your Mac, you can switch between cards by choosing "setup" from the "target" menu. A dialog will appear. The topmost pulldown menu is the "target." At this point, you can switch cards. Then click "OK" in the dialog and you will notice the message "M2 released from reset" in the Terminal window of the debugger. At this point, you must quit the debugger and re-launch it.
- 3DODebug will report a syntax error when trying to display a variable name that is also the name of a field in a structure.

4.4.2 Notes

- Contrary to what users might expect, the debugger's "H/W Reset" command does not reset the Tasks list. This means that if you use the "debug" command to debug or run a program, you should execute Reset before the program quits. Otherwise, the debugger thinks the program is still running. If you then try to run that program without re-executing the "debug" command, the debugger acts as if you had used the "debug" command anyway and spends time loading symbols and setting the initial breakpoint.

  This is not a bug; the debugger is behaving the way it is supposed to. What is at issue here is the name of the Task list window. Actually, the Task window displays a list of symbol files rather than an actual list of tasks.

4.4.2.1 Running the debugger in ROM mode

You may run your dev card in "ROM mode." In this mode, the dev card acts very much like a consumer unit. The debugger is not functional in ROM mode. You can, however, boot an M2 CD-ROM as it would boot on a real consumer unit.

To run in ROM mode:

- Run the `FlashROMTool`.
- In the window titled "`Onboard Flash`", click in the "`Filename`" box.
- Select the file "`ROM.m2.unenc`".
- Click the "`Flash`" button.
- When you see the "`Verified`" message, quit the `FlashROMTool`.
- Run the debugger. In the "`Target`" menu, select "`Setup`".
- Switch the "`ROM`" from "`Debugger`" to "`Flash`".
- Select the script "`flash.scr`".
- Click on "`OK`".
- The system will boot in ROM mode. You will not see any messages on the debugger console.

If you wish to switch back to debugger mode, you will need to use the `FlashROMTool` again to copy the file "`ROM.m2.debug`" into your dev card (see section 2.1.1 on page 2). Then in the debugger `Target/Setup` menu, switch the "`ROM`" back to "`Debugger`" and switch the script back to "`debugger.flash.scr`".

4.5 Comm3DO Application

4.5.1 READ THIS NOW!

Comm3DO App cannot be run simultaneously with 3DODebug.

4.5.2 Caveats and Known Bugs

- Certain makefiles, including Comm3DO when trying to compile the Comm3DO client task, have problems using Link3DO. The workaround is to remove the `-m2` option from the link3do line in the makefile.

4.6 CreateM2Make

4.6.1 READ THIS NOW!

Developers are expected to create new makefiles with each release of the software CDs.

4.6.2 Caveats and Known Bugs

- Don't create any makefiles that request optimization and debugging to be turned on simultaneously.
- Some compiler options have been removed, because they are now stored in the config file.
- The makefile now creates a "`<>.spt`" file along with the executable. This script tells 3DO Debug where to find source code and symbols. Note that this does not work with multiple source directories. See the 3DO Debugger manual for more information.
- On a PowerPC Macintosh, attempting to view source directories if they are set using a "`.spt`" file will crash the Macintosh with an illegal instruction error. Note that CreateM2Make automatically generates a "`.spt`" file.

  This is not a problem if you are using a 68k Macintosh or if you are viewing the directory when it is set manually.

5.0 OS

5.1 Kernel

5.1.1 Caveats and Known Bugs

- The debugging console view must currently always be opened full-screen due to a hardware bug.
- The operating system sometimes allocates memory in a way that causes fragmentation of the heap. This can create problems for applications that need large, contiguous memory allocations.

  A workaround is to preallocate large, contiguous allocations.

6.0 Graphics

6.1 READ THIS NOW!

- When linking a graphics program, the following link line should be used:

```
-lmusic -lframe2d -lframe_utils -lframework -lbsdf -lpbsdf -lframework
-lpipeline -lframe_utils -lbsdf -lpbsdf -lclt -lspmath -lfile
-leventbroker -lc
```

- Minimizing Footprint: to decrease your memory footprint, keep in mind the following:
  - use indexed textures whenever possible
  - make sure that textures, materials and models are shared wherever possible.
- Interfacing with C++

  The Graphics header files are designed for inclusion in C++ programs. In order to include the Graphics header files from C++ you should make sure that the preprocessor variable "GFX_C_Bind" is defined in your makefile.

6.2 Caveats and Known Bugs

- The instructions to run the "anim" program that are included in the header of the source file "anim.c" are out of date.

  To run anim, do the following:

  - Build the anim program using the default makefile or a makefile created by CreateM2Make.

    If you are using a new makefile created by CreateM2Make, move your "anim" executable (and ".spt" file) into the following folder:

    3DO:3DO_os:M2_2.7:Remote:Examples:Graphics:Frame:Anim

    The skeleton.csf file is already in the directory by default

    You can also leave the anim program where the CreateM2Make makefile placed it, and move or copy the skeleton.csf file needed to run the example into the same directory.
  - Launch 3DODebug and navigate to the directory containing the anim executable.
  - At the command line, type
    anim skeleton.csf skeleton_world

  The usage message that appears if you enter the command incorrectly is also out of date.

  The correct usage is:

  anim <filename>.csf <filename>_world
- The introduction to Chapter 5 of the *M2 Graphics Tools* manual contains a list of the graphics tools available in M2, along with a brief but useful explanation of what each tool is designed to do. Unfortunately, this useful and comprehensive list is difficult to find if you don't know exactly where to look for it because the manual's Table of Contents does not contain any reference to it.
- The footnote on Page 34 of the *M2 Graphics Tools* manual is incorrect and should be deleted. The M2 development hardware now supports compressed textures.
- In the *M2 Graphics Tools* manual Appendix D, on Page 255, the second typedef on the page (the typedef of the Seq_Action struct) and the warning note that follows should be changed to read as follows:

```
typedef struct {
    PackedID ChunkType;    /* 'ACTN' */
    uint32   dataSize;
    float    rate;         /* Frames per second */
    uint16   numPOV;
    uint16   numExposures; /* Number of frames referenced below */
    uint16   frameRefNums[];
    Seq_Exp  exposures[];  /* Optional */
} Seq_Action;

---------
Warning: The following chunk is currently not supported. It is
provided for discussion and example purposes only.
----------
```

6.3 Mercury

The texture lib, texture tools, gcomp program (called gmerc for Mercury), and converters have all been updated with new functionality and bug fixes to work with Mercury. The versions of these tools in the Mercury folder on the CD are more up to date than the versions in the Graphics folder.

6.3.1 READ THIS NOW!

- When using Mercury code, use the link3DO linker program contained in the Mercury Release folder.

  See section 4.3.1 on page 4 of this document for a more detailed description of the problems that can be encountered by using the regular link3DO with Mercury code.

- At this time, the Mercury rendering engine has not been completely integrated with all of the older graphic components. In this release, conflicting definitions in some of the Mercury header flies can cause problems when using fonts with the Mercury engine.

  The conflicting definitions are "`Color4`" in "`lighting.h`" and "`Point3`" in "`matrix.h`".

  You can avoid these problems by editing the two header files as follows:

  - In "`mercury/lighting.h`" add

    ```
    #ifdef MACINTOSH
    #include "graphics:gp.h"
    #else
    #include "graphics/gp.h"
    #endif
    ```

    and remove the "`Color4`" struct definition.

  - In "`mercury/matrix.h`" remove the following two lines:

    ```
    #define Point3 Vector3D
    #define pPoint3 pVector3D
    ```

6.3.2 Mercury Run-Time

6.3.2.1 Caveats and Known Bugs

- `Vector3D_GetX()`, `Vector3D_GetY()`, and `Vector3D_GetZ()` get a syntax error when compiling.

  Workaround: Remove the incorrect ending semicolon from three "define" statements in "`matrix.h`":

  Change this:

  define Vector3D_GetX(v) (v)->x;
  define Vector3D_GetY(v) (v)->y;
  define Vector3D_GetZ(v) (v)->z;

  to this:

  define Vector3D_GetX(v) (v)->x
  define Vector3D_GetY(v) (v)->y
  define Vector3D_GetZ(v) (v)->z

- `Matrix_Zero()` causes a crash when used.

  Workaround: Do not use this function. Instead, you can write your own code to zero out values in a matrix.

6.3.2.2 Fixed Bugs

- `Matrix_Mult()` now returns correct values.

6.3.3 Mercury Examples

6.3.3.1 Caveats and Known Bugs

- `helloworld` and `viewer` do not scale their input models. This makes the default model for these examples very small.
- In `helloworld` and `viewer`, setting the initial camera location to "0, 0, 0" causes a program exception. Avoid setting it to "0, 0, 0."
- `helloworld` and the `viewer` do not display models with transparency correctly because they do not initialize the source address field "srcaddr" of the current frame buffer. As a result, this field contains a random value.

  Workaround: Edit the example code to initialize the "srcaddr" field. Consult the *3DO M2 Mercury Programmer's Guide* for a description of how to do this.

- `viewer` does not calculate bounding boxes correctly. This may cause erroneous warnings that you have a zero-size bounding box.

  Workaround: Edit the example code, and insert your own code to calulate the bounding box.

6.3.3.2 Fixed Bugs

- Binary SDF models with a single pod no longer get a program exception when used with the viewer example.

6.3.4 Mercury Texture File Formats

As most M2 developers know, a UTF (unified texture format) file is an M2-specific file based on the IFF file format. It's an IFF format of the FORM type with the four-letter designator "TXTR".

Mercury users must also become familiar with the concept of a *UTF page file*. A UTF page file is simply a UTF texture file (that is, a FORM-type "TXTR" file) with an additional page chunk designated "M2PG". This chunk contains information about the sub-textures contained in the texel data chunk (now thought of as the texture page). M2PG is meant to be easy to parse and to be very general-purpose and not subject to change from release to release.

Additionally, there is the concept of a *Mercury page file*. This is a UTF page file with an additional chunk designated "PCLT". This chunk is very implementation-specific and will probably change as Mercury evolves. Think of this chunk as a compilation of all the data necessary to render a sub-texture within a texture page. This chunk is extremely hard to parse as it is mostly CLTs. Right now, the creation of this chunk is considered to be a one-way street and the M2 Texture Library and texture tools will merely skip over this chunk when they see it. When this chunk is present, the header chunk and TAB chunk are not needed and can be stripped out of the final texture. Future implementations of utfpage and the texture reader will eliminate the need of the DAB and M2PG chunks from the final textures as well, simplifying reading and processing of incoming textures.

IFF CAT files are simple concatenations of standard IFF FORMs. In this case, we are only interested in concatenations of "TXTR" FORMs, which are the UTF textures.

IFF LIST files are a more sophisticated way of storing multiple FORMs in a single file and allow for sharing of properties between FORMs. For example, an IFF LIST of UTF textures could contain twenty texture pages which share the same PIP. This would mean that only one copy of the PIP data ever need be stored in the file, thus saving the space required by nineteen PIP chunks. Because IFF LISTs are MUCH more complicated to parse (and may be too inefficient for a runtime environment) the current texture reader in Mercury doesn't support IFF LIST files. Adding IFF LIST support will be investigated and will become part of the runtime reader if it's performance is satisfactory.

6.3.5 Mercury Texture Tools

This section describes several Mercury tools and explains how they handle various file formats.

- `utfpage`

  This is the workhorse. It takes an ASCII texture file (TexArray and TexPageArray) as input and uses it to construct a Mercury page file. If multiple pages are specified, `utfpage` concatenates them into an IFF CAT file that can be read by the mercury reader.

  Features:

  Any Texture Application setting that is set in the TexArray or the texture itself is converted to a `CltSnippet` and becomes part of the texture. This includes settings such as *WrapModes*, *PIPSelect*, *Blend Operations*, and the like. It should be noted that the settings in the TexArray have precedence over those in the texture file (which are stored in the TAB chunk). One upshot of this is that it is no longer necessary to manually set the *WrapMode* settings of a texture before creating the page. All that is necessary is to set proper *WrapMode* in the TexArray (and this field is set by all the 3D converters provided by the Tools group) entry for that texture.

One of the input options to the utfpage tool is an external PIP. Why would you want to specify a custom PIP for page processing? To allow for each sub-texture within a page to have its own unique PIP. Because the TexArray entry for a sub-texture can specify a PIP Index Offset, it's possible to have a multiple PIPs represented within the page's PIP. For example, four six-bit textures have at most 256 colors total between them. That means that all four textures PIPs could be represented entirely with a single PIP that is a concatenation of all the PIP. To create this concatenated PIP, one can use *utfpipcat*. Then that PIP file can be passed on to *utfpage*. The only other necessary step is to set each texture's TexArray entry to have the appropriate *PIPIndexOffset*. The first texture will have a *PIPIndexOffset* of 0 (so it doesn't need to be set), the second will have an offset of 64, the third 128, etc. Rather than create a PIP by hand and manually compute the *PipIndexOffset* for each TexArray entry, one can use the -superpip option. This option takes all the subtextures' PIPs in a page and concatenates them into a single PIP for the page. The restriction that the total number of colors of all the PIPs cannot exceed 256 is still in effect. This is a quick way to make shared PIPs, but the result may not be optimal since each texture in the page now has it's own exclusive PIP. In most applications, many textures are bound to have a common PIP and these values could be shared. The *-superpip* option, as handy as it is, makes no attempt to find these.

`utfpage` has a new option, *-superpip*. This option is extremely useful for creating pages quickly without having to worry about have a single PIP for the whole page. This option concatenates each page's sub-textures' PIPs into a single PIP for each page. It also automatically sets the PIPIndexOffset for each sub-texture so that each texture will reference it's own unique PIP. The restriction is that the number of colors of all the sub-textures' PIPs cannot exceed 256. For example, if you have a bunch of 64-color textures, you could have four per page with the -superpip option. The sub-textures need not share the same bit depth or number of colors so you could have two 64-color textures and still have room in the PIP for 8 16-color textures. You could even have two eight-bit textures as long as their total number of color didn't exceed 256 (one could have 100 colors and the other 156, for example).

Limitations:

All the textures in a given page must share the same PIP (one way or another) and the sum of their texel data cannot exceed 16KB. Right now, each page uses the first valid PIP of a sub-texture to be the PIP for the whole texture UNLESS a PIP is specified in the command-line options (see above) or the -superpip option is used. If a PIP is specified in the command-line options, that PIP will be used for each texture page.

- `utfunpage`

  This recreates the original textures in a UTF page file, however, it currently only works on single page files. Texture page CAT files *(i.e.*, a file with multiple texture pages) must first be run through utfsplit. This will be changed shortly.

- `utfsplit`

  This takes UTF files which are either in CATs or LISTs (i.e. multiple UTF textures in a single file) and splits them into individual UTF texture files.

- `utfquantmany`

Takes a series of images and find a common pip for them all and does color reduction if necessary. There is an option to quantize the input textures (and overwrite them) to that resulting PIP.

- `utfpipcat`

  Concatenates the PIPs of several input files into a single PIP. Useful for have unique PIP among textures within a texture page. See below.

- `utfpipsub`

  Replaces the ssb and alphas in a PIP with one that the user specifies. Useful if you want all entries in a PIP to have say SSB set to 1.

6.3.5.1 Caveats and Known Bugs

- `quantizer` bug:

  The latest version of `quantizer` will crash upon exit on the Macintosh platform. `quantizer` will write out the proper quantized texture before crashing, and you can recover from the crash if Macsbug is installed.

  To work around this problem, you can use `utfquantmany` instead of `quantizer`. `utfquantmany` acts like a multiple-file version of `quantizer` that also performs the duties of `utfmakepip`.

  This bug can be eliminated from `quantizer` by commenting out the last two `M2TX_Free` calls in the source code (immediately after the `M2TX_WriteFile` call) and recompiling the tool.

  This bug is being fixed for a future release.

- `utffit` bug:

  `utffit` can be made to crash by doing one of two things.

  - Failing to provide in the argument list the number of LODs to generate.

    If the number of LODs to generate is omitted (a common usage mistake), `utffit` evaluates the next argument (usually a string) as a number and comes up with 0. A division by 0 error will eventually occur.

  - Giving `utffit` an impossible task.

    If you give `utffit` an image that won't fit into 16KB, given the conditions specified by the user, the program will also get a division by zero error when it attempts to resize the image to have a 0 dimension.

    For example, if `utffit` is asked to create an image that has 11 LODs, there is no way such an image can fit into 16K because, to have 11 LODs, the image must be at least 1024x1024 (2^10). `utffit` will try to downsize the image all the way down to 0x0 to create a 16K image, and this results in a division by 0.

    This bug is being fixed for a future release.

- If you enter the command "`ppmtoutf`" with no parameters in order to get usage information on a Macintosh or an SGI machine, you will not get a usage prompt.

  To stop the command, you must interrupt it, using Command-period on a Macintosh or Control-c or Delete on an SGI machine. Entering and interrupting the command as described here should not cause any problems for your system.

6.4 Graphics Folio

6.4.1 Features Added or Changed

This release completes the core functionality for the Graphics Folio. The new features in this release are as follows:

- Added non-interlaced Projector support. You can now create and display Views on non-interlaced displays.

  Consequently, two new Projector types now exist:

  - NTSC-nolace
  - PAL-nolace
- Added the support code required to let applications to switch between Projectors on the fly. No special cooperation is required between Tasks; the folio handles everything.
- Added new functions that let applications use and manipulate Projectors:
  - `ActivateProjector()` and `DeactivcateProjector()`

    These functions turn Projectors on or off, making their View hierarchy visible or invisible.
  - `NextViewTypeInfo()`

    Lets an application iterate through all the View types a Projector can support. You can use this to search the database intelligently for the View type most appropriate for the application.
  - `NextProjector()`

    Lets an application iterate through all the available Projectors in the system.
  - `QueryGraphics()`

    Makes available to applications some of the Graphics Folio's internal state information -- such as the current default Projector.
- You can now call `OpenItem()` and `CloseItem()` on a Projector. You have to do this if you wish to make a different Projector visible using `ActivateProjector()`.
- Added a new field to the Projector Item structure: `p_Flags`.

  This field currently reports whether or not the given Projector is active.
- Increased the priority of the VBL interrupt to 210 because it has to occur very early to effect display mode changes.
- Added two new flags to the `vti_Flags` field in the `ViewTypeInfo` structure:
  - VTIF_ABLE_AVG_H

    Set if the View can do horizontal pixel averaging.
  - VTIF_ABLE_AVG_V

    Set if the View can do vertical pixel averaging.
- Added a new error code to the folio: `GFX_ERR_PROJINACTIVE`.
- Added three new example programs in the folder "`3do_os:M2_2.7:remote:Examples:Graphics:GraphicsFolio`":
  - `listprojectors`

    Lists all the Projectors in the system.

- `listviewtypes`

  Lists all the View types in a given Projector.

- `manyview`

  Illustrates the creation, display, and manipulation of multiple Views at once.

- To document these changes, we made major changes and additions to graphics folio autodocs. Autodocs were written for the new functions, and the documentation for the Projector Item (listed under Portfolio Item Reference) was significantly amended.

  In particular, the Projector autodoc introduces the concept of a "default Projector," which is the Projector used when you do not explicitly specify one to the folio.

  READ THESE DOCUMENTS before using the new Projectors.

6.4.2 Caveats and Known Bugs

- There is a small performance degradation with double buffering. The workaround when performing benchmarking is to use single buffering.

6.5 Graphics State (GState)

6.5.1 Caveats and Known Bugs

- Passing `GS_FreeBitmaps()` an array of uninitialized bitmap Items will cause a "`Read from location 0 error`", which will cause M2 to crash. This is likely to happen if the application takes an early exit, calling `GS_FreeBitmaps()`, before `GS_AllocBitmaps()` has been called.

  Workarounds:

  - Make sure that when your application takes an early exit, your application determines whether `GS_AllocBitmaps()` has been called, and avoids calling `GS_FreeBitmaps()` if `GS_AllocBitmaps() has not been called.`
  - Your application can call `GS_FreeBitmaps()` with `numBitmaps = 0`.

6.6 Font Folio

- `SetClipBox()` Fails to Clip

  The `SetClipBox()` function, with the enable clipping flag turned on, fails to clip text.

  Workaround:

  Anytime after calling `SetClipBox()`, call `MoveText()` with the `dx` and `dy` parameters equal to zero. This will not move the text but will activate the clipping.

- Large Fonts May Display Incorrect Letters

  Some large fonts (larger than 24 points) can cause a character to be incorrectly rendered. This happens because the logic that determines how many characters can fit in the Texture RAM is off by one character, which causes the last character to overwrite the first character in TRAM. This problem is more likely to occur with larger fonts for which the size of the entire font character data is more than 16 kilobytes.

  Partial Workaround:

- Render the corrupted character in a separate TextState, or
- Use SetClipBox() to clip the target text. This often eliminates the incorrect character overwrite but also causes part of the text not to be displayed.

- When `DrawText()` or `DrawString()` are used in conjunction with other functions that use destination blending, such as some of the Frame2d functions, the rendered text may have cosmetic errors.

  This happens because the font folio does not initialize all of the destination blending registers.

  Workaround: Before calling `DrawText()` or `DrawString()`, call the CLT function `CLT_SetSrcToCurrentDest()` to set the source buffer equal to the current destination buffer.

  You must do this before each call or series of calls to these functions. You do not have to do it within a series of contiguous calls to one or both of them.

  This problem is currently being fixed for a future release.

- Font Folio problem with z buffering.

  Text strings with background color do not render when zbuffering is enabled with the CLT_DBZCNTL register set to (0, 0, 0, 0, 1, 1) because the background rectangle and the foreground text are both rendered with the same W value of 0.999998.

  (Note that the register setting "0,0,0,0,1,1" is the setting to update the zbuffer and render pixels with a smaller z value, which is the typical way to enable zbuffering.)

  There are 2 workarounds for this problem:

  - The preferred workaround:

    Set DBZ_DBZCNTL to (0, 0, 1, 1, 1, 1); i.e., use DBZCNTL (GS_Ptr(g), 0, 0, 1, 1, 1, 1). This setting updates the zbuffer and renders pixels with a smaller or equal z value.

  - The less desirable workaround:

    Render the text string without background color.

  In a future release, the PenInfo structure will be extended to allow the application to specify different W value for both the background and the foreground of text strings.

6.7 3D Graphics Libraries

6.7.1 Framework

6.7.1.1 Caveats and Known Bugs

- If you are using the built in shapes: cube, cylinder, torus, or sphere in an SDF file, the primitive will receive the wrong material index. This does not affect the SDF files created by our conversion tools, since they never use these primitives.
- If you cannot close a binary SDF file, you should make sure that you have recompiled your code with this 2.7 release.
- `Scene_SetTransparent` is not working correctly. Currently, setting scene transparency to FALSE turns it on, while setting it to TRUE turns it off. The default is still to have scene transparency off.

6.7.2 Pipeline

6.7.2.1 READ THIS NOW!

- Only one GP is allowed per application.

6.7.2.2 Caveats and Known Bugs

- You cannot use the scaling transforms with lighting, this will lead to an incorrectly illuminated object.
- There is a bug in the 3d pipeline which causes the hardware to hang when rendering faceted pre lit triangle surfaces.

  A temporary buffer is allocated for holding the transformed and clipped vertices, initially this is 32k (each vertex can occupy up to 36 bytes). At the start of each primitive the space needed to hold the vertices is compared against the current size and if necessary the buffer is reallocated, however no check is done in the 2.0 or 2.7 version of the software to ensure that this allocation was successful. If the system is running low on memory and the primitive is too large to fit in the current buffer then the machine will hang.
- The only time when a possibly valid return code would be interpreted as TXB_None (-1) is in the case where the user gets the DblDitherMatrixA and DblDitherMatrixB. Values of DitherMatrix are only valid if you actually set them. The user should ignore the non-error code.
- If you are using a macro in the Graphics Lib, do not treat it as a function; a macro to a pointer with ++ will lead to unforeseen results.

6.8 Command List Toolkit (CLT)

6.8.1 Caveats and Known Bugs

- There is a bug when using the Triangle Engine under 2.0 and 2.7. The TE is only reset the first time the triangle engine is used. This means that the first program you run that uses the TE will be running with the TE in a clean state. Subsequent programs will start with the TE left in the state of the last program executed. This means that you must reset all the TE registers to known values or risk unpredictable results.

6.9 Geometry Compiler

6.9.1 Caveats and Known Bugs

- The Geometry Compiler has problems dealing with extremely large files.
- The Macintosh version of the Geometry Compiler behaves unreliably in some environments and on some data files.

6.10 Modeling Package Converters

6.10.1 Strata (ssptosdf)

6.10.1.1 Caveats and Known Bugs

- There is a bug in Strata's texture mapping dialogue; it shows up when you use default textures. You can set horizontal and vertical field "%'s", but the number may or may not be output. The work around is to go back and retype a "%" in the field a second time. Change it and hit ok; otherwise you get a zero value and you will get weird texture coordinate number values or your software will crash.

This bug is being fixed for a future release.

6.11 PostPro

6.11.1 READ THIS NOW!

This version does not allow preview of 3D models on the 3DO M2 hardware.

6.11.2 Caveats and Known Bugs

- PostPro only writes the first level of detail when doing a conversion from texture to RGB.
- Texture files with multiple LODs will be shifted to the right by about a quarter inch when saved to a new name from PostPro.
- If PostPro's memory is low, conversions from a texture may result in garbled images. It doesn't take much to max out the PostPro's memory. Therefore, if you are getting garbled images, try increasing PostPro's partition size.

7.0 Video and Data Streaming

7.1 READ THIS NOW!

- Because the audio clock is generated by the CD-ROM clock, the CD-ROM power must be on (and the red indicator light on) in order for audio output, audio timers, and Data Streaming to work. If a program is stuck, try turning on the CD.

7.2 Data Streaming Run-Time Libraries

7.2.1 Data Stream run-time library

7.2.1.1 READ THIS NOW!

- The client application should NEVER call `DSSetPresentationClock()`. (Applications used to call `DSGetClock()` to work around a DataStreamer bug.)

7.2.1.2 Caveats and Known Bugs

- The trace logging facility (for real-time debugging) is set to print its logs to the debugger's Terminal window.

  To write the logs to files instead, set the compile-time switch `USE_RAW_FILE` in `subscribertraceutils.c` to "1".
- `DSGoMarker(...,GOMARKER.BACKWARDS)` will return a range error if the stream is beyond the last marker point. This bug is being fixed for a future release.

7.2.2 Subscriber library

7.2.2.1 SAudio Subscriber

Caveats and Known Bugs

- This subscriber supports multiple channels but, because of the limitation of 23 signals per thread, do not use more than four channels simultaneously.
- The sound spooler allocates a signal for each sound buffer allocated. When multiple channels are active at the same time, multiple sound spoolers are created which may cause the 23 signals per thread limitation to be exceeded.

To avoid this, reduce the number of audio buffers allocated per sound spooler by

- Specifying the number of audio buffers as four (the default is eight) when chunkifying your audio data:

```
AudioChunkifier -cs 4 -i sample_audio.aiff -o sample_audio.saudio
```

or by

- Changing the number of buffers using the SAudioTool on a chunkified audio file:

```
SAudioTool -nb 4 -i sample_audio.saudio
```

7.2.2.2 MPEG Audio Subscriber

READ THIS NOW!

An MPEG audio chunk holds one compressed audio frame, so it typically consists of only 1274 bytes including chunk header -- depending on the compression bit rate chosen. The streamer needs a subscriber message for each chunk that is outstanding at a subscriber, plus a couple more per subscriber for other requests. Otherwise, the streamer will run out of subscriber messages and abort stream playback.

Consequently, to play a stream file containing only MPEG audio chunks, either decrease the total stream buffer space (e.g. 20K bytes/buffer * 4 buffers should be plenty), or increase the number of subscriber messages to a few more than the number of chunks that will fit in the total stream buffer space at the same time.

7.2.2.3 EZFlix Subscriber

READ THIS NOW!

- Programs that use the EZFlixSubscriber must link in the EZFlix Decoder library: `libezflixdecoder.a`.

Caveats and Known Bugs

- This subscriber uses an older method of processing, in which it does not begin decompressing a frame until it is almost time to display it. This can cause a delay before the image actually appears on the screen.

7.3 Data Streaming Examples

7.3.1 PlaySA

7.3.1.1 Caveats and Known Bugs

- For MPEG audio stream files, PlaySA can only play all channels simultaneously.

7.3.2 VideoPlayer

7.3.2.1 Caveats and Known Bugs

- This app does not turn on horizontal or vertical interpolation.

  This app ought to use the `BMTAG_BUMPDIMS` tag and the bumped bitmap dimensions instead of the hard-coded dimensions.

- If the image is smaller than 320x240, it places it in the upper left-hand corner of the display rather than centering it.

7.3.3 EZFlixPlayer

7.3.3.1 Caveats and Known Bugs

- When using MemDebug, you may get the following error message when using EZFlixPlayer:

  ```
  Data Access Fault
  ```

  This is caused by a memory access bug in the clean-up code of `EZFlixPlayer.c`. This bug is being fixed for a future release.

7.4 Data Streaming Tools

7.4.1 AudioChunkifier

7.4.1.1 READ THIS NOW!

- Though the AIFC formats provide less compression than MPEG Audio, their run-time decompression consumes virtually no PowerPC CPU cycles, because decompression takes place in the 3DO M2 DSP (or in other 3DO M2 HW in the case of SQS2). MPEG Audio decoding, however, will consume a substantial portion -- 25% or more -- of the CPU.

7.4.1.2 Caveats and Known Bugs

- Currently supports input formats with the following choices of parameters, although the 3DO M2 Audio Folio does not support all possible parameter combinations (see "format restrictions" below). Also, testing priority has gone to 16-bit formats, which we expect to be more widely used than 8-bit formats for reasons of fidelity. See SquashSnd documentation and its usage message for more information on the AIFC compression options.
  - 44100 or 22050 Hz sample rates;
  - stereo or mono;
  - 8 or 16 bit per sample quantization;
  - uncompressed (AIFF) or one of 3 compressed (AIFC) formats:
  - SQS2: 2:1 compression; decompression with M2 HW;
  - CBD2: 2:1 compression; decompression with M2 DSP SW;
  - ADP4: 4:1 compression; decompression with M2 DSP SW.
- Format restrictions:
  - SQS2 compression applies to mono only. (CBD2 works for both mono and stereo.)
  - ADP4 compression applies to mono only.
- SQS2 vs. CBD2 for mono: since SQS2 decoding is HW-based, it takes almost no DSP resources (ticks or program space), but CBD2 may produce slightly better audio fidelity, and CBD2 can be used for stereo.
- ADP4 currently may not be an optimal ADPCM implementation, due to the level of audio distortion perceptible in some material.

7.4.2 MPEGVideoChunkifier

7.4.2.1 Caveats and Known Bugs

- If you omit the `-fps` (frames per second) command-line option, MPEGVideoChunkifier will assume the output frame rate should be that which is specified in the video sequence header of the input MPEG-1 Video

Elementary Stream. This is an enumerated parameter, for which only the following frame rates are defined by the MPEG standard: 23.976, 24, 25, 29.97, 30, 50, 59.94, or 60 fps. Use the `-fps` option to force MPEGVideoChunkifier to make the output frame rate a different value. It will neither add nor subtract frames from the input file, it will only make their frame rate different. This feature is necessary, for example, if you want to use Sparkle to encode QT movies with non-MPEG-standard frame rates (e.g., 12, 15, 20 fps), and if you want to perform the encoding in such a way as to preserve the non-standard frame rate. In this case, you must know the non-standard frame rate of the MPEG Video stream, and specify it to the MPEGVideoChunkifier explicitly, using the `-fps` option.

- MPEGVideoChunkifier is a fairly easy-to-use tool for weaving simple, linear, stand-alone MPEG movie clips. For such simple streams, MPEGVideoChunkifier provides adequate defaults for most of its input parameters, and you need not learn about the details. But for preparing more complex streams, e.g., preparing separate clips which will later be concatenated, or preparing branch points, it is important to understand the command-line parameters in detail.

7.4.3 MPEGAudioChunkifier

7.4.3.1 Caveats and Known Bugs

- Known bug: Invoking MPEGAudioChunkfier with `"-help"` or incorrect arguments will print usage, which may be missing the command name, or may print garbage where the command name should be printed.

7.4.4 EZFlixChunkifier

7.4.4.1 READ THIS NOW!

- Using the `"-h"` and `"-w"` options to crop a movie can cause a crash. This bug is being fixed for a future release.

  Workaround: This bug does not happen if all frame dimensions are multiples of 16.

7.4.4.2 Caveats and Known Bugs

- You must specify the desired frame size using the `"-h"` and `"-w"` options or else frame size 256x192 may be assumed.

  Note that all frame dimensions must be multiples of 16 because of the aforementioned bug.
- The `"-k"` option, to flag key frames, has been removed.
- The `"-q"` option allows you to set the quality level. Use 25% for an Opera level of quality. Use 75% if you can accept the higher data rate.
- On PowerPCs, garbage characters may be seen in the verbose output when altering the height and width making the output file unusable.

7.4.5 QTVideoChunkifier

7.4.5.1 Caveats and Known Bugs

- QTVideoChunkifier only accepts an EZFlix-compressed QuickTime movie file as input. Since EZFlix is the only SW codec currently supported by 3DO M2 Data Streaming, it's the only type of QT movie file you should be using as input.

- Known bug: The QTVideoChunkifier experiences problems on the PowerMac due to an incompatibility between the EZFlix QT Component 1.0b1 and the QuickTime™ 2.0 component. To resolve this problem, either use a utility like "Thing Motel" to install the EZFlix Component, or use a more recent version of QuickTime™ (2.1 or later).

7.4.6 DATAChunkify -- the DATASubscriber's Data Prep Tool

Caveats and Known Bugs

- This tool DOES NOT do data compression. If the `-comp` option is used, the data is assumed to have been compressed with the comp3DO MPW tool, and the chunk header is set so that the DATASubscriber will decompress the data when it is read from the stream.

7.4.7 Weaver

7.4.7.1 Caveats and Known Bugs

- Weaving a stream containing lots of small chunks (e.g., 1 KByte or smaller) with a stream containing fewer large chunks (e.g., half the streamblocksize or larger) can result in a stream where the large chunks fall increasingly behind the small chunks, according to their timestamp order.

  This bug does not occur when the larger chunk type can be split across streamblock boundaries.

  However, when large chunks cannot be split (e.g. EZFlix video), smaller chunks of other streams (e.g. audio) can still move ahead of them.

  If enough small chunks move far enough ahead to make playback unsmooth or blocked up (you can verify this with DumpStream), increase the audio chunk size (e.g. to 4 KBytes or more).

  This problem might be severe if EZFlix video were used with MPEG audio, but that combination is not recommended because both those decoders require big percentages of the CPU cycles.
- To set the starting presentation time of an MPEG video elementary stream, use the MPEGVideoChunkifier's "`-s`" option, NOT the relative starting time in the Weaver-script "`file`" command.

  Always set the relative starting time in the Weaver-script "`file`" command to "0" for an MPEG video chunk file. The "`file`" command's relative starting time argument can only adjust the delivery timestamps (DTSs), not the MPEG video presentation timestamps (PTSs). Attempting to use it to adjust MPEG PTSs would make playback unsmooth and unsynchronized.
- Weaver scripts should ALWAYS include the "`writestreamheader`" command. Many of the parameters specified by the Weaver script cannot be rendered effective unless the Weaver creates a stream header containing them. It will only do so if this command is included in the script.
- The Weaver's "`writemarkertable`" command adds a spurious first marker, which points to a chunk in the first stream block, e.g. the third header chunk. (Unlike all other markers, this one is not followed by a FILL chunk out to the end of the stream block.) Having an extra marker affects some of the go-marker run-time operations.
- The "`writegotochunk`" command only works when its options argument is "1". It won't accept additional option flags or alternative branch types.

7.4.8 Worksheet and Exercises

On the 3DO M2 2.7 CD, you will find source code for the four Data Streaming example applications, as well as example scripts and data for chunkifying and weaving playable streams. See the folder with pathname:

`Examples:M2_2.7:Streaming:`

Within this folder, in addition to the four source-code folders, you will find a text file named "`ds_examples_readme`" and a folder named "`Tools_and_Data`".

The `readme` file explains the example apps and streams contained in the overall folder.

The `Tools_and_Data` folder contains an MPW worksheet named "`Video.worksheet`", which shows how to use the various chunkifying and weaving tools to create playable streams. `Tools_and_Data` also contains various elementary streams (MPEG-Video, MPEG-Audio, AIFF audio, and EZFlix video) for use with the worksheet examples.

7.4.8.1 READ THIS NOW!

- The installer will NOT install this folder on your Mac, because of the large size of some of the files.

7.5 Video Tools

7.5.1 3-2 Pulldown, MovieEdit, MovieCompress

7.5.1.1 Caveats and Known Bugs

- MovieEdit: If you attempt to save a movie from the same directory that has the same name as a movie that is also open, both movies will end up invalid. The work around is to close the original movie before replacing it.

7.6 MPEGVideo

7.6.1 Features Added or Changed

- The M2 MPEG Video device no longer clears the sequence header information and state after receiving a flush or detecting a bitstream error. This allows an application to continue decoding from a stream which does not contain frequent sequence header information.
- After detecting a real bitstream error (most likely caused by a media read problem), the M2 MPEG Video device seeks forward to the next picture in the input stream before attempting further decode. This helps to speed recovery.

7.6.2 Bugs Fixed

- In MPEG video driver, fixed a bug in `NextStartCode()` search that was affecting the Video CD player. The start code search algorithm has been changed to handle cases where start codes cross certain buffer boundaries.
- Spurious MPEG hardware interrupts are generated under certain conditions (e.g., after video bitstream DMA and "picture done" interrupts occur at close to the same time). The device's interrupt handler now handles these interrupts correctly.

  This bug and the previously described search bug had been creating symptoms that appeared to be caused by errors in the video bitstream, including "bitstream error" messages.

7.7 MPEG Audio

7.7.1 Bugs Fixed

- The following bug was fixed in the MPEG Audio Decoder folio: If the input packet (or chunk) contained multiple "frames" (MPEG audio access units), the decoder used to emit the same Presentation Time Stamp (PTS) multiple times, which would cause jerky video when the video subscriber tried to sync to the audio. This fix is needed for the VideoCD application, and it will also help if the MPEG AudioChunkifier is ever modified to emit more than one frame per chunk.

8.0 Audio

8.1 Caveats and Known Bugs

- Boards below the Rev G level will not work with M2 Release 2.0 or 2.7.
- Envelope time scaling does not affect envelope loop times set by `AF_TAG_SUSTAINTIME_FP` and `AF_TAG_RELEASETIME_FP`.
- `SampleInfoToTags()` fails to transfer `smpi_Detune` to an `AF_TAG_DETUNE` tag. This also affects `LoadSample()`. The result is that any detune information stored in an AIFF is not copied to the resulting Sample Item.

8.2 Beep Folio

8.2.1 Caveats and Known Bugs

- When using the Beep Folio, you should turn down the amplitude on channels that are stopped. This will prevent DC offsets from those channels from being added to your output which could cause clipping.
- In M2 Release 2.0 or 2.7, if you use the Beep Folio and then want to use the Audio Folio, you must reboot the 3DO Dev Card before switching to the Audio Folio. If you use the Audio Folio and then want to use the Beep Folio, you must reboot the 3DO Dev Card before switching to the Beep Folio.

  This will be fixed in a later release, which will support demand loading and unloading of folios.

8.3 Audio folio

8.3.1 Caveats and Known Bugs

- There is a known bug with velocity zones. In order for velocity zones to work properly, a pitch must be specified along with the velocity when calling `StartInstrument()`. Specify the pitch using `AF_TAG_PITCH`.

8.4 Music Library

8.4.1 Sound spooler

8.4.1.1 Caveats and Known Bugs

- Worked around the sample length alignment checks in the audio folio, which could cause perfectly good buffers to be rejected. However, this workaround can permit illegal buffer lengths to be played by the audio folio. A more

reliable system to prevent buffer length alignment problems is being investigated.

Appendix A Examples in Tools Folder

Below is a list of the examples that are shipped with individual tools.

A.1 Audio Examples

A.1.1 test3sf

Source code showing how to call the 3SF sfPlayScore API from a 3DO program. Located on the CD, inside ":Tools:M2_2.7:Audio:3SF:Examples:".

A.2 Development Code Examples

A.2.1 Comm3DOExampleClient 68K

Source code showing how to call the Comm3DO API from a Macintosh program.

Located on the CD, inside
":Tools:M2_2.7:LowLevel:Comm3DO:Example Client App:"

A.2.2 Comm3DOExampleClient PPC

Source code showing how to call the Comm3DO API from a Macintosh program. Located on the CD, inside
":Tools:M2_2.7:LowLevel:Comm3DO:Example Client App".

A.2.3 C3 Task

Source code showing how to call the Comm3DO API from an M2 program. Located on the CD, inside
":Tools:M2_2.7:LowLevel:Comm3DO:Example Client Task:".

A.3 Graphics Examples

This is a listing of example and sample code that is to be found in individual tools folders. The path name for these examples is
":Tools:M2_2.7:Graphics".

A.3.1 M2 Kanji FontViewer

- **KFontViewer <fontName>**

  Demonstrates how to use the Kanji font library to load and display S-JIS encoded text on the 3DO system.

Appendix B Examples in Examples Folder

The 3DO M2 CD contains a central "Examples" folder that contains examples for various M2 components. This folder is in the following location:

```
3do_os:M2_2.7:remote:Examples
```

A list follows of all the example programs in the "Examples" folder.

All the example programs are documented by man pages contained in the *3DO M2 Supplemental Portfolio Reference* unless otherwise noted.

Note that some of the locations given in the man pages are incorrect. The locations given in the following list are correct.

B.1 READ THIS NOW!

- Makefiles for examples have debugging turned on.
- Makefiles should include self as a dependency.

B.2 Audio Examples

The audio examples are contained in

```
3do_os:M2_2.7:remote:Examples:Audio
```

B.2.1 Beep (in Audio:Beep)

- **tb_envelope**

  Trigger envelopes using the Beep folio.
- **tb_playsamp**

  Play a sample using the Beep Folio.
- **tb_spool**

  Spool audio from memory using the Beep folio.

B.2.2 Juggler (in Audio:Juggler)

- **tj_canon**

  Uses the juggler to create and play a semi-random canon.
- **tj_multi**

  Uses the juggler to play a collection.
- **tj_simple**

  Uses the juggler to play two sequences.

8.4.2 MarkovMusic (in Audio:MarkovMusic)

- **MarkovMusic**

  Constantly-varying soundtrack. that responds to an environmental influence defined by your application -- in this case the control pad.

B.2.3 Misc (in Audio:Misc)

- **capture_audio**

  Record the output from the DSP to a host file.
- **minmax_audio**

Measures the maximum and minimum output from the DSP.

- **playsample**

  Plays an AIFF sample in memory using the control pad.
- **simple_envelope**

  Simple audio envelope example.
- **ta_attach**

  Experiments with sample attachments.
- **ta_customdelay**

  Demonstrates a delay line attachment.
- **ta_envelope**

  Tests various envelope options by passing test index.
- **ta_pitchnotes**

  Plays a sample at different MIDI pitches.
- **ta_sweeps**

  Demonstrates adjusting knobs.
- **ta_timer**

  Demonstrates use of the audio timer.
- **ta_tuning**

  Demonstrates custom tuning a DSP instrument.
- **tone**

  Simple audio demonstration.

B.2.4 Score (in Audio:Score)

- **auto_beat**

  Automatic rhythm demo that uses many AIFF library samples.
- **playmf**

  Plays a standard MIDI file.

  Note that playmf is a shell command and is documented in the chapter of the *3DO Supplemental Portfolio Reference* that describes shell commands.
- **playpimap**

  Automatic rhythm demo that uses lots of AIFF library samples.

B.2.5 Sound3D (in Audio:Sound3D)

- **ta_bee3D**

  Three sounds in a navigable space using 3DSound API.
- **ta_leslie**

  Demonstrates directional sound with the 3DSound API.
- **ta_sound3d**

  Simple directional sound using 3DSound API.
- **ta_steer3d**

  Control a walking man in three-space using the 3DSound API.

- **SeeSound**

  Sounds in an audio-visual space, using 3DSound and Framework/Pipeline.

B.2.6 SoundEffects (in Audio:SoundEffects)

- **sfx_score**

  Uses libmusic.a score player as a sound effects manager.

- **windpatch**

  Creates a "wind" sound effect using the audio folio's patch compiler.

B.2.7 Spooling (in Audio:Spooling)

- **playsf**

  Demonstrates how to loop a sound file off disc.

- **ta_spool**

  Demonstrates the libmusic.a sound spooler.

- **tsp_algorithmic**

  Advanced sound player example showing algorithmic sequencing of sound playback.

  Caveats and Known Bugs

  When running the audio examples tsp_algorithmic, tsp_rooms, tsp_spoolsoundfile, and tsp_switcher, you may notice some stuttering and popping. If you see this behavior, try putting the debugger in Special Mode (Command - =). This bug appears to be a result of the hard drive not delivering data fast enough. Special Mode improves I/O speed and is a more accurate representation of actual CD data access.

- **tsp_rooms**

  Room-sensitive soundtrack example using advanced sound player.

- **tsp_spoolsoundfile**

  Plays an AIFF sound file from a thread using the advanced sound player.

- **tsp_switcher**

  Advanced sound player example that switches between sound based on control pad input.

B.3 Event Broker Examples

The Event Broker examples are contained in

```
3do_os:M2_2.7:remote:Examples:EventBroker
```

- **cpdump**

  Queries the event broker and prints out a summary of what's connected to the control port.

- **focus**

  Talks to the event broker and switches the focus to a different listener.

- **lookie**

  Connects to the event broker and reports any events that occur.

- **luckie**

Uses the event broker to read events from the first control pad.

- **maus**

  Uses the event broker to read events from the first mouse.

B.4 FileSystem Examples

The FileSystem examples are contained in

```
3do_os:M2_2.7:remote:Examples:FileSystem
```

- **ls**

  Displays the contents of a directory.

- **type**

  Type a file's content to the output terminal.

- **walker**

  Recursively displays the contents of a directory, and all nested directories.

B.5 Graphics Examples

The Graphics examples are contained in

```
3do_os:M2_2.7:remote:Examples:Graphics
```

B.5.1 CLT (in Graphics:CLT)

- **cltspinclip**

  Demonstrates various basic CLT features.

  Notes

  - Upon exiting the CLTspinclip program a spurious error message is displayed because the program is trying to free a signal that was never allocated. This error does not affect the running of the program and can be ignored. This bug is being fixed for a future release.

- **cltsurface**

  Uses the CLT to create and display a 3D surface.

- **gsquare**

  This example covers: initialization of the graphics folio and gstate, creating and setting bitmaps, double buffering, loading and display of textures, and rendering geometry, including rectangles, strips and fans.

- **height_field**

  Displays a 3-D terrain section whose texture is determined by its height.

  This example is self-documenting. Execute it with no parameters to see usage message.

- **scrolling**

  Demonstrates multi-level parallax scrolling with the CLT.

B.5.2 Fonts

8.4.2.1 Fixed Bugs

- The following bug was fixed:

  The font example makefiles ("`fonts.make`" and "`showmessage.make`") located in the directory:

"M2_2.7:3DO:Examples:M2_2.7:Graphics:Fonts:" do not work. They are backlevel versions and will terminate during the build.

8.4.2.2 Fonts Examples (in Graphics:Fonts)

These examples are self-documenting. Execute them with no parameters to see usage message.

- **bounce**

  Shows how to set up a TextState, move, scale, and read a TextState's current position by bouncing the text around the screen.

- **colors**

  Shows how to change text colors.

- **newfont**

  Runs through a series of font folio tests, and times all the tests.

- **rotate**

  Shows how to rotate text on screen.

- **showfont**

  Display characters of a font.

- **showmessage**

  Outputs any words on the command line of the debugger to the display monitor.

- **testnl**

  Tests whether the font folio correctly handles \n characters.

B.5.3 Frame (in Graphics:Frame)

- **anim**

  Displays object with keyframe animation using Framework extensions.

- **freespin**

  Freely rotate specified objects within a binary SDF.

- **m2perf**

  Program to measure the performance of Framework, Pipeline, and the M2

- **newview**

  Rotate specified objects within a binary SDF using control pad.

- **sceneperf**

  Indicates performance data for rendering a scene.

- **testspin**

  Rotate objects within a binary SDF and record performance statistics.

  This example is self-documenting. Execute it with no parameters to see usage message.

B.5.4 Frame2d (in Graphics:Frame2d)

- **automapper**

  Demonstrates how to map the corners of a sprite.

- **drawlines**

Demonstrates three methods of drawing lines.

- **movesprite**

  Shows how to move, scale, and rotate a sprite

- **points**

  Shows how to draw points.

- **rectangles**

  Demonstrates two methods of drawing rectangles.

- **renderorder**

  Shows how to alter the render order of sprites using a list-based approach.

- **simplesprite**

  Very simple example of how to draw a single sprite.

- **spin3d2d**

  Shows how to display 3D and 2D graphics simultaneously.

B.5.5 GraphicsFolio (in Graphics:GraphicsFolio)

- **autosizeview**

  Illustrates how to create a Bitmap and View automatically sized for the prevailing video mode.

- **basicview**

  Illustrates how to create a basic View.

- **listprojectors**

  Lists all the Projectors in the system.

- **listviewtypes**

  Lists all the View types in a given Projector.

- **manyview**

  Illustrates the creation, display, and manipulation of multiple Views at once.

B.6 Kernel Examples

The Kernel examples are contained in

```
3do_os:M2_2.7:remote:Examples:Kernel
```

- **defaultport**

  Demonstrates using a task's default message port to communicate with a child thread.

- **fasttiming**

  Demonstrates how to use the high accuracy kernel timing services.

- **memdebug**

  Demonstrates the memory debugging subsystem.

- **metronome**

  Demonstrates how to use the metronome timer feature.

  msgpassing

  Demonstrates sending and receiving messages between two threads.

- **signals**

  Demonstrates how to use signals.

- **timerread**

  Demonstrates how to use the timer device to read the current system time.

- **timersleep**

  Demonstrates how to use the timer device to wait for an amount of time specified on the command-line.

- **UserExceptions**

  Demonstrates how to trap program exceptions.

B.7 Miscellaneous Examples

The Miscellaneous examples are contained in

```
3do_os:M2_2.7:remote:Examples:Miscellaneous
```

- **compression** (in `Miscellaneous:Compression`)

  Demonstrates use of the compression folio.

- **dbgconsole** (in `Miscellaneous:DbgConsole`)

  Example program used to demonstrate 3DODebug functionality.

- **numinfo** (in `Miscellaneous:DebugExamples`)

  Example program used to demonstrate 3DODebug functionality.

B.8 MPEG Examples

The MPEG examples are contained in

```
3do_os:M2_2.7:remote:Examples:MPEG
```

- **photo** (in `MPEG:photo`)

  This example application shows how to use the MPEG-Video decoder interface for still-image decompression.

  The example loads an MPEG-Video still-image (I-frame) from a file into M2 memory. The file format is simply an MPEG-1 Video elementary stream containing only an I-picture. (It can be created, for example, using Sparkle.) The Photo app then opens the MPEG-Video device for I-picture-only decoding, so that the device allocates no reference-frame buffers in main memory. The device returns the decompressed I-frame to the Photo app as soon as it is finished decoding, without the pipeline delay incurred with full I/P/B movie decoding.

- **playfast** (in `MPEG:playfast`)

  Demonstrates the power of the 3DO M2 MPEG-1 Video decoder, and also serves as example code for the MPEG-1 Video device driver.

  Works by loading into 3DO M2 RAM as much of an input MPEG-1 Video elementary stream as it has space for, and then playing this stream repeatedly in a loop.

  This example does not use the Data Streaming interface to M2's MPEG-1 Video decoder.

B.9 Data Streaming Examples

This section summarizes and gives pointers to the Data Streaming example applications. For more detail and notes on most of these applications, see "Data Streaming Examples" on page 18.

The Data Streaming examples are contained in

```
3do_os:M2_2.7:remote:Examples:Streaming
```

- **DataPlayer** (in `Streaming:DataPlayer`)

  Uses the DATASubscriber to "play" a stream containing DATA chunks. The application simply prints statistics about data blocks as they are delivered, but it exemplifies how to set up, interact with, and destroy the DATASubscriber.

- **EZFlixPlayer** (in `Streaming:EZFlixPlayer`)

  Plays an EZFlix (3DO M2 SW-decodable video) stream, with or without a synchronized native 3DO M2 audio stream.

- **PlaySA** (in `Streaming:PlaySA`)

  Plays one or more native 3DO M2 audio streams woven into a single multiplexed stream.

- **VideoPlayer** (in `Streaming:VideoPlayer`)

  Plays an MPEG-1 Video stream, with or without a synchronized native 3DO M2 audio stream.